# ASP.NET API

# Project Set Up

## Setup Steps for Visual Studio

From within VS (using point and click):

* Create new solution called UsingHighLevelAppFrameworks
* Add a new ASP.NET Core Web API project called ASPNETAPI-demo.csproj to the solution
* Use NuGet to add a reference to the Newtonsoft.json JSON package

## Setup Steps for Visual Studio Code

From Command Prompt:

* Prepare solution by typing:

dotnet new solution --name using\_high\_level\_app\_frameworks

* Create webapi project, and add to solution by typing:

dotnet new webapi --output aspnet\_api\_demo

dotnet sln add aspnet\_api\_demo

* Install JSON package in the aspnet\_api\_demo project by typing:

dotnet add json-demo package newtonsoft.json

## Now do the following regardless of project type:

* Note the project contains some ready written code that pretends to generate some weather information.
* Configure the app to expose port 6001 when browsed to via https by editing line 17 of the launchSettings.json file located in the project’s Properties folder:

...

"applicationUrl": "https://localhost:6001;http://localhost:5001",

...

* Review the code and try to work out what it’s doing.
* Build and run the code and then open a browser window and enter the following URL:

https://localhost:6001/Weatherforecast

* You should see a textual response that includes randomly generated weather forecasts for the next five days.

**Edit the app to include code that exposes a Movies API**

* Right click on the Controllers folder and add a new Controller class called MoviesController.cs (For Visual Studio select Add, Controller… and then API Controller – Empty from the dialog. For VSC select New C#, API Controller)
* Add the following code to the MoviesController class:

[HttpGet]

public string Get()

{

return "Hello from the Movie API";

}

[HttpPost]

public string Post([FromBody] string title)

{

return $"You submitted the film called {title.ToUpper()}";

}

// Using query parameters, via https://localhost:6001/film/movieData/ET/Steven%20Spielburg/1982

[HttpGet]

[Route("movieData/{title}/{director}/{releaseYear}")]

public string GetMovieData(string title, string director, int releaseYear)

{

return $"The film {title} was released in {releaseYear} and directed by {director}";

}

// Collecting information from URL path, via https://localhost:6001/film/movie/22

[HttpGet]

[Route("movie/{id}")]

public object GetMovie([FromRoute] int id)

{

//TODO: Add code that interogates Movie DB for movie with the passed in id

var movie = new

{

Title = "Star Wars",

Director = "George Lucas",

YearReleased = 1977,

Revenue = 775800000

};

return movie;

}

* Review the code and try to work out what it’s doing.
* Build and run the code:
  + Visual Studio: Clicking the green triangle (F5) will automatically launch a browser window accessing the site via Swagger.
  + VSC: dotnet run --project aspnet\_api\_demo and then open a browser window and enter the following URL :

https://localhost:6001/Swagger/index.html

* You should see the following window:

Graphical user interface, text, application

Description automatically generated

* Clicking on each of the drop-downs allows you to test each of the methods. Confirm they all work as expected.

**Edit the app to include some additional methods that pass parameters in alternative ways**

* Add the following to the foot of the MoviesController class:

// Receive and return headers

[HttpPost]

[Route("headers")]

public string PostHeaders([FromHeader]int MovieId)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

sb.AppendLine("You submitted headers...");

string additionalData = string.Empty;

foreach (string header in Request.Headers.Keys)

{

sb.AppendLine($"- {header} = {Request.Headers[header]}");

if (header == "movieid")

{

//TODO: Code to look up movie from Id

Response.Headers.Add(header, Request.Headers[header]);

Response.Headers.Add("MovieTitle", "Star Wars");

additionalData = $"Returning header:\n{header}: {Request.Headers[header]}";

additionalData += $"\nMovie Title: Star Wars";

}

}

sb.AppendLine(additionalData);

return sb.ToString();

}

// Extract payload as strongly-typed JSON object

[HttpPost]

[Route("movie")]

public string PostNewMovie([FromBody] Movie movie)

{

return $"{movie.Title} was directed by {movie.Director} " +

$"it was released on {movie.ReleaseDate} and grossed {movie.Revenue}";

}

}

public class Movie

{

public string Title { get; set; }

public string Director { get; set; }

public DateTime ReleaseDate { get; set; }

public decimal Revenue { get; set; }

}

* Review the code and try to work out what it’s doing.
* Build and run the code:
  + Visual Studio: Clicking the green triangle (F5) will automatically launch a browser window accessing the site via Swagger.
  + VSC: dotnet run --project aspnet\_api\_demo and then open a browser window and enter the following URL :

https://localhost:6001/Swagger/index.html

* Test the newly added methods via the Swagger interface and confirm they are working as expected.